AI Integration of BlockShield Project

Final AI Engine: Addressing Data Challenges and Building the Model

Nashat Alfarajat – BlockShield AI Development Team

CrossRealms - intern

About the Data

The dataset used in this work was assembled from a combination of provided and publicly available PCAP files. The initial files — "old_honeypot_capture1.pcap" and "old_honeypot_capture2.pcap" — were supplied at the start but mostly contained normal background traffic. Additional captures, namely "honeypot_capture1.pcap", "honeypot_capture2.pcap", "honeypot_capture3.pcap", and "honeypot_traffic.pcap", were later collected, though they also did not provide significantly better coverage of malicious activity. Finding PCAPs with the specific traffic types of interest is generally difficult, but to strengthen the dataset an external file — "2025-01-04-four-days-of-scans-and-probes-and-web-traffic-hitting-my-web-server.pcap" from Malware-Traffic-Analysis.net — was added. All seven PCAPs were then merged into a single dataset for subsequent processing and analysis.

PCAP File Parsing and Flow-Level Feature Extraction (CICFlowMeter-Style) Using Scapy

To prepare the honeypot network captures for AI model development, the raw .pcap files were parsed into flow-level features using a custom Python script built with Scapy. The goal was to emulate the feature extraction style of CICFlowMeter, producing statistical and protocol-aware attributes for each bidirectional network flow.

Approach

1. Packet Reading – Each .pcap file was read using scapy.rdpcap().
2. Flow Identification – Flows were defined by a 5-tuple: (Source IP, Destination IP, Source Port, Destination Port, Protocol), with reverse traffic aggregated into the same flow record.
3. Direction Separation – Packets were categorized as forward or backward to capture asymmetric behaviors.
4. Feature Computation – For each flow, statistical metrics were calculated, including:
    - Packet size stats: sum, mean, max, min, standard deviation
    - Inter-arrival times (IAT): mean, max, min, std, total
    - Flow duration, packets per second, bytes per second
    - TCP flag counts: FIN, SYN, RST, PSH, ACK, URG, CWE, ECE
    - Active/idle time statistics
    - Bulk transfer stats: average bytes/bulk, packets/bulk, bulk rate
    - Header length, segment size, and down/up ratio
5. Data Output – The resulting features were compiled into a Pandas DataFrame, where each row represents a single flow.

This process transforms raw packet captures into a structured, feature-rich dataset suitable for downstream machine learning or AI-based intrusion detection. From approximately **244,820 raw packets**, the parsing stage produced **70,621 flow records** with 88 features each. At this stage, the dataset is unlabeled, meaning there is no ground-truth classification of flows as malicious or benign. Nevertheless, it captures a comprehensive view of both traffic behavior and protocol-level characteristics, providing a solid foundation for subsequent labeling and for training AI models to detect malicious activity more effectively.

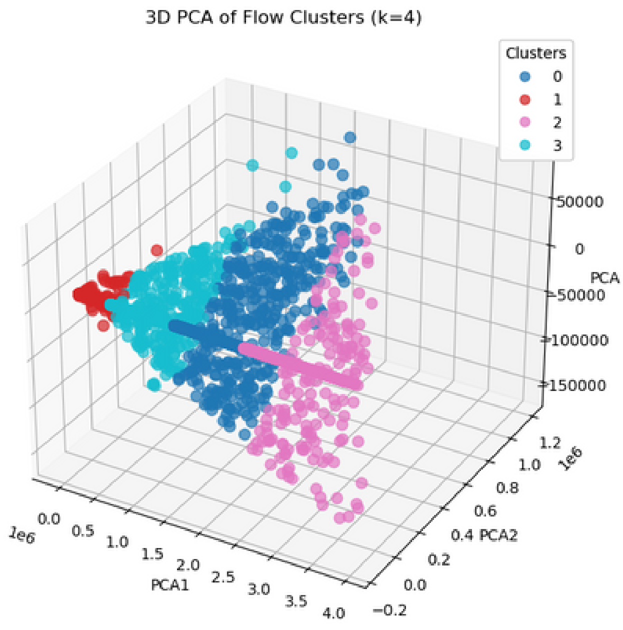
Clustering-Based Label Assignment

Feature Selection: A subset of flow-level features was selected to capture essential traffic behavior and protocol characteristics. The chosen features included:

- Flow statistics: Flow Bytes/s, Flow Packets/s, Flow Duration
- Packet size metrics: Min/Max Packet Length, Packet Length Mean, Fwd Packet Length Max/Min/Mean, Bwd Packet Length Mean/Max, Average Packet Size
- Temporal metrics: Flow IAT Max, Fwd IAT Max, Idle Mean
- Traffic ratio: Down/Up Ratio

Feature Scaling: Several normalization methods were evaluated to prepare the features for clustering. Standard scaling resulted in low silhouette scores, while applying a power transformer to mitigate skewness further degraded both the silhouette score and the Dunn index. Ultimately, Robust Scaler was chosen as it provided more stable results across the selected features.

Clustering Algorithm: Density-based methods (DBSCAN and HDBSCAN) were tested but produced an excessively large number of clusters, which limited their interpretability for this dataset. In contrast, k-means clustering with k = 4 yielded a strong separation of flows, achieving a high silhouette score of 0.96. Given this performance, k-means was selected as the final clustering method for assigning pseudo-labels to the flows.



The K-Means clustering produced four distinct groups with a silhouette score of 0.96 , indicating reasonably well-separated clusters. Based on feature patterns and domain knowledge.

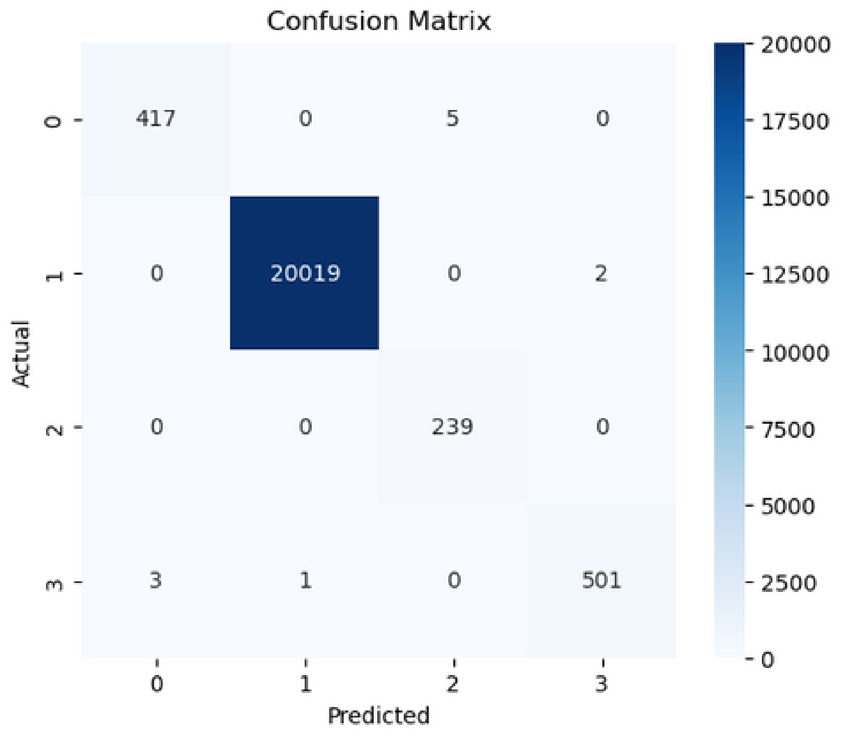
Cluster Interpretation: Interpreting the resulting clusters was challenging and the reliability of the assignments is limited, but general traffic patterns can still be observed. Cluster 1 is the most distinct, representing the only group of flows with notable throughput (≈50 bytes/s, higher packet rates, larger packet sizes, and shorter durations), suggesting active communication traffic. In contrast, Clusters 0, 2, and 3 all exhibit extremely low throughput, long durations, and very consistent small packet sizes. These clusters are best described as background or idle connections, likely reflecting long-lived sessions, keep-alives, or system-level noise rather than meaningful activity. The separation of clusters 0, 2, and 3 appears to be driven mainly by differences in flow duration and idle times, rather than by fundamentally different behaviors. Overall, the clustering mainly distinguishes between active traffic (Cluster 1) and various forms of low-activity/background noise (Clusters 0, 2, and 3).

Note: The 3D PCA plot looks messy due to dimensionality reduction, which is expected.

Training a Model Using Cluster-Based Labels

The merged and clustered dataset was first split into training and testing subsets, and all features were scaled using StandardScaler to ensure consistent contribution to the model. Because the cluster-based pseudo-labels were highly imbalanced, SMOTE (Synthetic Minority Oversampling Technique) was applied to the training set to generate synthetic samples for underrepresented clusters. This balancing allowed the model to learn effectively from all clusters without bias toward the dominant class, providing a more equitable foundation for supervised training based on the cluster assignments.

A Random Forest model was trained on the oversampled training dataset generated with SMOTE, while testing was performed using only the original, unaltered data. The model achieved excellent performance on the test set, demonstrating strong generalization despite the initial class imbalance. Specifically, the test results were: accuracy of 0.9995, macro-averaged precision of 0.9921, macro-averaged recall of 0.9950, and macro-averaged F1-score of 0.9935, indicating that the model was able to reliably predict the cluster-based pseudo-labels across all classes.



Conclusion

The dataset used in this project presented significant challenges. The quality of the provided honeypot captures was low, dominated by background traffic with little meaningful malicious activity, and obtaining suitable publicly available PCAPs with the desired traffic types proved difficult. These limitations affected the ability to build robust models directly from the data. Given these constraints, a more reliable alternative for future work would be to focus on publicly available, labeled datasets for supervised learning, which could provide higher-quality data and more generalizable results than attempting to work with limited or noisy honeypot captures. Another potential approach is to leverage the existing processing and clustering pipeline developed in this project: if higher-quality PCAPs become available in the future, the pipeline could be applied to generate labeled flows and train improved models without starting from scratch.